Published in: Proceedings 12th International Conference on Scientific and Statistical Database Management
Berlin, Germany, July 26-28, 2000, IEEE Computer Society Press

Generating Network-Based Moving Objects

Thomas Brinkhoff
Institute of Applied Photogrammetry and Geoinformatics (IAPG)
Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven (University of Applied Sciences)
Ofener Str. 16/19, D-26121 Oldenburg, Germany
tbrinkhoff@acm.org, http://www.fh-oldenburg.de/iapg/personen/brinkhof/index.html

Abstract

Benchmarking spatiotemporal database systems requires the generation of suitable datasets simulating the typical behavior of moving objects. Previous approaches do not consider that in many applications the moving objects follow a given network. In this paper, the most important properties of network-based moving objects are presented. These properties are the basis for specifying and developing a new generator for spatiotemporal data. This generator combines a real network with user-defined properties of the resulting dataset. A framework for using and promoting the generator exists.

1 Introduction

One of the most challenging applications of state-of-the-art technology is the field of traffic telematics, which combines telecommunication and computer science in order to establish traffic information and assistance services. Many current and future traffic telematics services need the management of large sets of spatial objects moving within a street network [1].

The support of motion requires that the database system efficiently organizes moving objects. Therefore, the management of spatiotemporal data is one of the most vivid research activities in these days [5]. One important task is the preparation and use of well-defined test data and benchmarks enabling the comparison of data structures and algorithms designed for spatiotemporal database systems.

In this paper, which is a short version of [2], the generation of spatiotemporal data is based on a network. The presented requirements to the generation of spatiotemporal data are based on the experience of the author while he was working for a company in the field of traffic telematics. However, it is not the goal of the paper to demonstrate (e.g. in a case study) that the data computed by the presented generator exactly fulfill the requirements of a specific application. Instead, a framework is presented where the user defines the exact behavior of the generator; such an approach allows considering the characteristics of a wider range of applications.

2 Related work

The experimental investigation of spatiotemporal database systems is a rather new field. Moving objects are represented by a set of instances consisting of one object identifier *obj.id*, several locations $obj.loc_i$ and additional time stamps $obj.time_i$.

For investigating spatiotemporal access methods, Theodorides, Silva, and Nascimento [8] have proposed the GSTD algorithm. Starting point is an arbitrary distribution of the objects. These objects are moved (and their size are changed) by using random functions, which can be parameterized. The result are sets of objects looking a little bit like moving and shape-changing clouds. In order to "create more realistic movements", Pfoser and Theodorides [3] have introduced a new parameter for controlling the change of direction. In addition, they use rectangles for simulating an infrastructure; each moving object has to be outside of the rectangles.

An application modeling fishing boats motivated Saglio's and Moreira's work [4]. They stress, "real-world objects do not have chaotic behaviors". Therefore, the objects have to observe the environment around them. Furthermore, they introduce object types, which allow modeling different behavior. In order to generate smoothly moving objects, objects of one class may be attracted or repulsed by objects of other types.

None of the above approaches uses a network for restricting the movement of objects.

3 The behavior of moving objects

A network often channels traffic. This observation holds for street traffic as well as for other means of transport. In consequence, almost no traffic can be observed outside of a network: ***(1) Moving real-world objects often follow a network***.

In the most cases, a moving object tries to use the fastest path to its destination. Therefore, it uses a path, which is the fastest or not far away from the fastest path: ***(2) Most moving objects use a fast path to their destination.***

Typically we find connections of different types in a network. In a street network, e.g., we find different street classes. The class of a connection has direct influence to the motion of objects: 1. The number of objects moving on a connection depends on this class; a low-class connection restricts the number of objects to a low value whereas a high-class connection attracts a considerably higher number of objects. 2. The speed of the objects is influenced by the class of the connection: high-class connections generally allow a higher speed than low-class connections. Also a speed limit in a country can be modeled by the classification of the connections. ***(3) Most networks consist of classified connections, which have impact on the number and on the speed of moving objects.***

The speed and number of moving objects are also influenced by other moving objects. Traffic jams are a common experience: If the number of vehicles using a street exceeds a threshold, the maximum and the average speed of the objects will decrease. The threshold depends on the capacity of the connection, which is in general dependent on the class of the

connection: ***(4) The number of moving objects influences the speed of the objects if a threshold is exceeded. This threshold depends on the class of the connection.***

The traffic also influences the paths of the objects. An example are detours which a driver may use if a traffic jam occurs: ***(5) Other moving objects influences the path of a moving object if the speed on a connection decreases.***

There exist further impacts on the number of moving objects: For example, rush hours depend on the time of day and the day of week. Holiday periods are another example. ***(6) The number of moving objects is a time-dependent function.***

Special weather conditions may decrease the maximum speed of (some) moving objects. Extended moving spatial objects allow modeling such effects: ***(7) The speed of moving objects is also influenced by a spatial or spatiotemporal function, which is independent of the network and of the objects moving on the network.***

The maximum speed of a moving object especially depends on its class. For example, private cars generally have a higher maximum speed than trucks. The maximum speed of trucks depends on their weight category. Note that other restrictions (e.g. statement 3) may cause that an object will never reach its speed limit. ***(8) Moving objects belong to a class. This class restricts the maximum speed of the object.***

The eight statements presented in this section describe the behavior of moving objects. They are motivated by the requirements of applications from field of traffic telematics. More motivating examples from other fields are presented in [2]. These statements surely do not consider all possible aspects of the movement of objects, but the use of a network requires observing the eight statements.

4 The network-based generation of data

According to statement 1, moving objects tend to follow a predefined network. Therefore, the basic data structure for generating spatiotemporal objects should be a network. In principle, it is possible to generate a synthetic network. There exist several reasons to use a real network. First, it is not a simple task to generate a realistic network. Second, the use of a network from a real-world application supports the generation of objects, which behave like the objects in this application. And third, several providers offer real-world networks; examples are the SEQUOIA 2000 Storage Benchmark [6] and TIGER/Line files [7].

4.1 The modeling of time

One important aspect for generating spatiotemporal objects is the modeling of time. First, we assume that a time period T restricts the existence of a spatiotemporal object. T is bounded by a minimum time t_{min} and a maximum time t_{max}. In real life, time is continuous. However, for the generation of spatiotemporal objects we have to assume that the time period T is divided by tn+1 time stamps t_i into tn time intervals $[t_i, t_{i+1})$. These parameters are globally used for all objects in order to simplify the generation process.

4.2 Requirements to the network

Each edge belongs to one edge class *edgeClass(edge)* and for each edge class a maximum speed *edgeClassMaxSpeed (edgeClass)* is (user-)defined (statement 3). The actual maximum speed on an edge *edgeMaxSpeed(edge)* has an individual value equal or less than the maximum speed according to the class of the edge:

edgeMaxSpeed(edge,time) ≤
edgeClassMaxSpeed(edgeClass(edge))

Furthermore, for each edge class a maximum capacity *edgeClassCapacity(edgeClass)* is user-defined. If the number of objects traversing an edge during a time interval *edgeUsage (edge,time)* is greater than the maximum capacity of the edge class, the maximum speed on the edge is restricted by a further limit described by a function *deceleratedSpeed* (statement 4):

edgeMaxSpeed(edge,time) ≤ *deceleratedSpeed(edgeClass(edge), edgeUsage(edge,time))* ,
if edgeUsage(edge,time)) > edgeClassCapacity(edgeClass(edge))

The dependence between *deceleratedSpeed* and its parameters is user-defined. Each edge includes a constant attribute *edgeClass* and a varying attribute *edgeUsage*. Furthermore, each edge has a time-independent spatial location *loc*.

4.3 Requirements to the moving objects

Each moving object belongs to an object class *objClass (obj)* and for each object class a maximum speed *objClassMaxSpeed(objClass)* is (user-) defined (statement 8). The speed of an object on an edge *objSpeed(obj,edge,time)* is restricted by the maximum speed of its object class and the maximum speed on the edge:

objSpeed(obj,edge,time) ≤ *objClassMaxSpeed(objClass(obj))*
objSpeed(obj,edge,time) ≤ *edgeMaxSpeed(edge,time)*

Each object requires an (non-changing) attribute *objClass*.

4.4 External objects

We can distinguish different types of external objects required by statement 7:

- External objects, which exist over the whole time, and others, which are created at *time1* and deleted at *time2*.
- Static and moving external objects.
- External objects with a static shape and external objects, which change their spatial extension over the time.

Basic properties of external objects are their position and extension *area(extObj,time)*. It will be assumed that a rectangle describes this area. The percentage, by which the speed in the area should decrease *(decreasingFactor)*, depends on the class of the external object. Now, a further limit to the maximum speed on an edge can be determined:

edgeMaxSpeed(time,edge) ≤
edgeClassMaxSpeed (edgeClass(edge)) ·
minimum ({decreasingFactor(objClass(extObj))
with intersection(loc(edge),area(extObj,time)) <> Ø*})*

4.5 Computing the motion

According to statement 6, the number of moving objects is a time-dependent function. In order to fulfill this requirement, the creation of new moving objects is controlled by a function *numberOfNewObjects(time)*, which computes the number of new objects for a time stamp. A moving object "dies" when it arrives at its destination or after the maximum time is reached.

The moving objects should follow a network (statement 1). Therefore, it is necessary to determine a starting node from the

network. In the following, *the network-based approach* is presented for computing starting nodes; other approaches can be found in [2]. The idea is to select a node by using a one-dimensional distribution function. Assuming a uniform distribution, each node is selected with the same probability. In this case, the distribution of the starting nodes is correlated to the density of the network.The starting node of a moving object is computed by the function *computeStartingNode(time)*.

An obvious approach to compute the destination of a moving object is to use the network-based approach again. However, this leads to non-satisfying results: A computation of the destination independently of the starting node leads to an uncontrolled distribution of the lengths of the paths. However, in real applications this distribution is not arbitrary but correlates to distinct regularities. In cities for example, short drives predominate. The (average) length of routes often correlates to the type of the moving object; e.g. trucks often have longer drives than small private cars. Another influencing factor may be the starting time. Therefore, the length of a route is determined by a user-defined function *computeLengthOfRoute(time,objClass)* in a first step. In the second step, the destination node has to be determined by the function *computeDestinationNode(time, startingNode,length)*.

Each object *obj* moves between two time stamps t_i and t_{i+1} from its actual position $obj.loc_i$ to its new position $obj.loc_{i+1}$ according to its computed route. In general, these positions do not match with the nodes of the network. The generator logs these positions. For the edges traversed between two time stamps t_{i-1} and t_i, the number of traversing objects *(edgeUsage)* is decremented and for the edges traversed now, this number is increased.

According to statement 2, the route computation should try to find a fast path to the destination following the connections of the given network. This can be achieved by using a traditional routing algorithm considering the resulting speed of an object *(objSpeed)* by weighting the edges.

Until now, the approach assumes that the speed of an object on an edge does not change between the starting and reaching the destination. However, this assumption contradicts the statements 4 and 5. One solution to solve this contradiction is to compute a new route at each time stamp. However, such an approach would require a huge computing power for the generation of moving objects and should therefore not be considered as a good solution. Furthermore, that approach is not realistic for a real-world application. In general, we can distinguish two situations where a new "computation" of a route is triggered: 1. by an external event (e.g. a message of the radio traffic service) or 2. by a strong deviation of the actual speed from the expected speed (e.g. if the car is in a traffic jam). In both cases, the moving object (or its driver) may compute a different route if some time has passed since the last computation. In order to simulate this behavior, two boolean functions are introduced: The user-defined function *computeNewRouteByEvent (time, timeOfLastComputation)* allows simulating external events. In a simple version it may return "true" if *time* minus *timeOfLastComputation* is larger than a given threshold. Another function *computeNewRouteByComparison (time, timeOfLastComputation, actualSpeed, expectedSpeed)* allows simulating the reaction in the case of a strong deviation between the actual speed and the expected speed.

5 The framework

The generation of network-based spatiotemporal data requires the following steps:

1. Loading the network from simple binary files. A tool exists which allows converting TIGER/Line Files [7].
2. The definition of the required user-defined functions and parameters as they are described in section 4.
3. The computation of the objects and their moves.
4. The report of the generated data into user-defined text files. In order to simplify the definition of the user-defined functions and parameters, the generator supports an ad-hoc visualization of the generated data.

We have built up a Java-based framework for performing these four steps. The web site *http://www.fh-oldenburg.de/iapg/personen/brinkhof/generator.html* has been prepared which allows downloading the generator, its documentation, network files and examples for user-defined functions and parameters. An applet, which can be used via the Internet, demonstrates the use of the generator and visualizes the computed moving objects.

Performance tests have shown that the generator computes large data sets within a reasonable time using a Java interpreter on a standard personal computer (for details see [2]).

6 Conclusions and future work

In this paper, a new generator for spatiotemporal data was presented. This generator combines a real network with user-defined properties of the resulting dataset.

Future work consists of the support of time-scheduled connections and of 3D-objects. Also the web site for promoting the generator should be developed further.

7 References

[1] Brinkhoff T.: "Requirements of Traffic Telematics to Spatial Databases". *SSD 1999*, Hong Kong, LNCS 1651, Springer, pp. 365-369.

[2] Brinkhoff T.: "A Framework for Generating Network-Based Moving Objects". Technical Report of the IAPG, FH Oldenburg/Ostfriesland/Wilhelmshaven, May 2000, *http://www.fh-oldenburg.de/iapg/personen/brinkhof/TBGenerator.pdf*

[3] Pfoser D., Theodoridis Y. : "Generating Semantics-Based Trajectories of Moving Objects". *Intern. Workshop on Emerging Technologies for Geo-Based Applications*, Ascona, 2000.

[4] Saglio J.-M., Moreira J.: "Oporto: A Realistic Scenario Generator for Moving Objects", *DEXA Workshop on Spatio-Temporal Data Models & Languages*, Florence, 1999, pp. 426-432.

[5] Sellis T.: "Research Issues in Spatio-temporal Database Systems". *SSD 1999*, Hong Kong, LNCS 1651, Springer, pp. 5-11

[6] Stonebraker M., Frew J., Gardels K., Meredith J.: "The SEQUOIA 2000 Storage Benchmark". *SIGMOD 1993*, Washington, DC, pp. 2-11.

[7] US Government Information & Maps Dep.: "TIGER/Line Files". *http://govdoc.ucdavis.edu/MapCollection/tiger.html*

[8] Theodoridis Y., Silva J.R.O., Nascimento M.A.: "On the Generation of Spatiotemporal Datasets". *SSD 1999*, Hong Kong, LNCS 1651, Springer, pp. 147-164.